ClusPath: a temporal-driven clustering to infer typical evolution paths

Marian-Andrei Rizoiu, Julien Velcin, Stéphane Bonnevay, Stéphane Lallich

Data Mining and Knowledge Discovery manuscript No.
(will be inserted by the editor)

ClusPath: A Temporal-driven Clustering to Infer Typical Evolution Paths

Marian-Andrei Rizoiu · Julien Velcin · Stéphane Bonnevay · Stéphane Lallich

Received: date / Accepted: date

Abstract We propose ClusPath, a novel algorithm for detecting general evolution tendencies in a population of entities. We show how abstract notions, such as the Swedish socio-economical model (in a political dataset) or the companies fiscal optimization (in an economical dataset) can be inferred from low-level descriptive features. Such high-level regularities in the evolution of entities are detected by combining spatial and temporal features into a spatio-temporal dissimilarity measure and using semi-supervised clustering techniques. The relations between the evolution phases are modeled using a graph structure, inferred simultaneously with the partition, by using a "slow changing world" assumption. The idea is to ensure a smooth passage for entities along their evolution paths, which catches the long-term trends in the dataset. Additionally, we also provide a method, based on an evolutionary algorithm, to tune the parameters of ClusPath to new, unseen datasets. This method assesses the fitness of a solution using four opposed quality measures and proposes a balanced compromise.

Keywords detection of long-term trends · evolutionary clustering · temporal clustering · temporal cluster graph · semi-supervised clustering · Pareto front estimation.

1 Introduction

Knowledge is often hidden in plain view, within the sheer amount of available data. Refining data into information by discovering patterns is the main purposes of Data Mining. In this paper, we are interested in the more specific problem of discovering general temporal trends, also known as typical evolution paths. This makes the problem of pattern mining more difficult, by adding to it the temporal dimension of data. It changes the definition of the learning problem, since the description of entities is temporally contextualized. We study a population of entities, described over a period of time by low-level descriptive features. Our final

Marian-Andrei Rizoiu
NICTA & Australian National University, 7 London Circuit, Canberra, Australia.
E-mail: Marian-Andrei.Rizoiu@nicta.com.au

Julien Velcin · Stéphane Bonnevay · Stéphane Lallich
ERIC laboratory, Université de Lyon, Lyon, France.
E-mail: Julien.Velcin@univ-lyon2.fr, Stephane.Bonnevay@univ-lyon1.fr,
Stephane.Lallich@univ-lyon2.fr

aim is to detect the typical paths of evolution taken by most of these entities over the extent of recorded time. Considering that an entity's description can change over time, we define an *evolution phase* as a period of limited extent in time during which multiple entities in the dataset share similar descriptions. Therefore, to be informative, evolution phases should be coherent in time and in the descriptive space. An *evolution path* is defined as a succession of evolution phases followed by a large number of entities in the dataset, which can also be seen as a typical trajectory through the evolution space. Such general evolution trends can reveal information about the more complex hidden phenomena which happen in the population of entities. We show, in Sect. 4, that high-level concepts such as socio-political regimes or fiscal strategies can be detected from the low-level descriptive features. For example, the "Swedish Social and Economical Model" (Erixon, 2000) can be mapped on the evolution path followed by the northern European countries, in a dataset described using features such as debt-to-GDP ratio, unemployment rate and political coloring of the parliament. Similarly, we detect a strong trend in a population of companies, in which significantly less tax is payed, while the net income increases: the fiscal optimization of international corporations.

To this end, we propose ClusPath, an algorithm that organizes a set of observations into a structured partition, coherent both in the descriptive space and in the temporal space. We use a temporal-aware dissimilarity measure for assessing the similarity between observations. Furthermore, we use a semi-supervised technique using must-link pairwise constraints to ensure the contiguous segmentation of observations associated to an entity. The resulted clusters are interpreted as evolution phases. The main novelty of ClusPath is that the three components of the clusters: i) descriptive, ii) temporal and iii) the graph of relations between them are inferred simultaneously, in one optimization procedure. This creates an intertwining, which allows the three components to influence each other, during the optimization process. The major advantage over other approaches (like co-clustering or post-clustering structure inference) is that it allows more flexibility during the optimization process and creates a partition more adequate to describe the data. Furthermore, it ensures that the entities have a smooth passage between the phases on the evolution path. While we run the risk of losing non-smooth entity evolutions, the purpose of our application is to detect the evolution paths followed by the majority of the population. This is due to the "slow changing world" assumption, which states that the long-term trends detectable in a population have a higher inertia and they evolve more slowly than entities. This assumption might not be desirable for all contexts, for example i) in applications in which it is important to capture the fined-grained evolutions of entities. (*e.g.*, stock exchange market) or ii) in which the general trend does not possess a high inertia (*e.g.*, popularity in social networks, online memes *etc.*). For these applications, ClusPath has a parameter (*i.e.* λ_2, defined in Sect. 3.2) which allows the modulation of the degree in which this hypothesis is enforced. Additionally, we propose an optional method, based on an evolutionary algorithm, to automatically tune the parameters of ClusPath on new, unseen datasets.

The remainder of this paper is structured as follows. In Sect. 2, we present some previous related work. In Sect. 3, we define the learning objectives, we translate them into an optimization problem, we introduce ClusPath and the evolutionary heuristic for automatically tuning the values of parameters. Sect. 4 presents the datasets we use, the quality measures and the performed experimentations. We conclude in Sect. 5 and plan some future work.

2 State of the Art

The purpose of *Evolutionary clustering* is to capture the temporal evolution of clusters, given data observations and their creation time. A good clustering result should fit the current data well, while simultaneously not deviate too dramatically from the recent his-

tory (Chi et al, 2007). Initial frameworks have been designed for distance-based clustering, such as K-Means and agglomerative hierarchical clustering (Chakrabarti et al, 2006). Chi et al (2007) have extended the evolutionary framework to spectral clustering with the emphasis on smoothing clustering results over time to avoid sudden changes. Xu et al (2012) take a generative models approach and apply it to dynamic social network analysis. All of these evolutionary clustering approaches rely on time discretization into temporal windows of arbitrary size, whereas ClusPath integrates the temporal dimension as a variable, without the need of discretization.

TDCK-Means (Rizoiu et al, 2012) was introduced to detect typical evolution phases in a population of entities. The main contributions of this work were i) proposing a temporal-aware dissimilarity measure, used to assess the similarity between two observations, both in the multidimensional descriptive and temporal spaces and ii) assuring a contiguous segmentation by imposing semi-supervised must-link constraints (Wagstaff et al, 2001) and a continuous time-dependent penalty function for breaking the constraints. Other algorithms in the literature use constraints for segmentation purposes. tcK-Means (Lin and Hauptmann, 2006) uses must-link constraints and inflicts a fixed penalty when the following conditions are fulfilled simultaneously: the observations are not assigned to the same cluster and the time difference between their timestamps is less than some threshold. Similarly, De la Torre and Agell (2007) detect tasks performed during a day, based on video, sound and GPS data. Aligned Cluster Analysis (Araujo and Kamel, 2014) is an extension of the kernel k-means clustering algorithm, in which side information is added in the form of pairwise constraints to its objective function. Its purpose is segmenting time-series and clustering them together.

A recent extension of TDCK-Means (Rizoiu et al, 2014) proposes an *a posteriori* method for organizing the constructed clusters as a graph. The construction is based on the transitions of entities between phases. While the aim of ClusPath is also to identify the links between clusters, it differs fundamentally from the *a posteriori* construction by inferring the relations *simultaneously while clustering* of the observations. This creates an intertwining, by allowing the partitioning to influence the structure of clusters, and, conversely, the links between clusters influence the assignment of observations to clusters.

ClusPath infers the relations between clusters by combining multiple criteria into an objective function and optimizing it using a gradient descent method. A related learning problem is *relational multicriteria clustering*. The aim is to detect clusters of alternatives in a multicriteria context and to identify relations between these clusters. In Rocha et al (2013), a classical clustering is first applied to the set of possible alternative and each cluster is evaluated using predefined measures. A partial order outranking relation is established between clusters, based on the scores of the evaluation measures and the preferences of the decision maker. The outranking relations are constructed as a *post-treatment* (after clustering), using a multi-criteria pair-wise comparison procedure. De Smet and Eppe (2009) use a distance measure that is based on binary preference relations between different alternatives. The distance is extended to construct a binary outranking matrix between clusters. The outranking matrix is constructed at each clustering iteration, but it has no influence over the assignment of actions to clusters and it is calculated solely based on the composition of clusters.

A distinct, but somewhat related field is that of clustering of multi-dimensional trajectories. The crucial difference between this field and our work is that the former usually seeks to find similarities between entire multi-dimensional data series (*e.g.*, storm path trajectories, drug therapy response) in order to find connections between the evolution of different entities. These approaches usually treat the entire temporal series as single data points. For example, Gaffney and Smyth (1999) model the set of trajectories as individual sequences of points generated from a finite mixture model. Liang et al (2013) predict glaucoma evo-

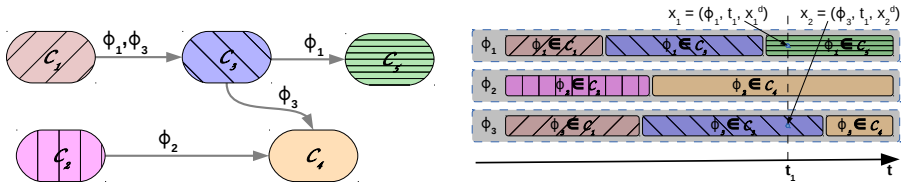
lution in patients by using previous recorded disease evolutions. In the first step, clustering is applied to gather patients similar to the target patient. The second step fits a predictive model on the set of patients found in the first step, and predicts the future disease condition. In (Siddiqui et al, 2012), a mixture model of Markov Chains is learned and used to predict the next most likely state/cluster per object. Apart from serving a different purpose, the first approach is unsupervised, the second and the third are supervised, whereas ClusPath is a semi-supervised algorithm. Kalnis et al (2005) approach a related learning problem: detecting of trajectories of moving clusters. Their underlying assumption is that the data contains dense groups of individuals which move together in space and time (*i.e.*, the moving clusters). They construct individual partitions at each timestep and detecting pairs of clusters in successive timesteps, susceptible of belonging to the same moving cluster. Our problem differs mainly because an evolution path is not a unitary entity as a moving cluster. The individual evolution phases have arbitrary extents of time and each one can be part of multiple evolution phases. Unlike the temporal instantiations of a single moving cluster, evolution phases have meaning by themselves and they are loosely connected in evolution paths.

3 Our Proposal

3.1 Formalization and learning objectives

Definitions and intuitions. Each studied entity $\phi_l \in \Phi$ is described using multiple attributes, which form the multidimensional description space. To each entity correspond N observations $(entity, timestamp, description)$. An observation $x_i = (\phi_l, t_m, x_i^d), i \in 1..N$ means that the entity ϕ_l is described by the vector x_i^d at the moment of time t_m. To identify the entity associated with a particular observation x_i, we use the notation x_i^ϕ. Therefore the notations x_i^ϕ and ϕ_l both denote the entities, and we use one or the other depending on the point of view (*i.e.*, observation- or entity- oriented). Similarly, x_i^t is the timestamp associated with the observation x_i. Our learning problem, starting from such a dataset, aims at detecting typical evolution phases and evolution paths. There is a double interest: a) obtaining a broader understanding of the phases that the collection of entities went through over time (*e.g.*, detecting the periods of global political instability, economic crisis, wealthiness *etc.*); b) constructing the trajectory of an entity through the different phases (*e.g.* a country may have gone through a period of military dictatorship, followed by a period of wealthy democracy). We define an evolution phase $\mathscr{C}$ as a set of observations x_i, so that observations belonging to $\mathscr{C}$ are as similar (in terms of a similarity function) as possible among themselves and dissimilar to observations in other phases. Unlike in classical clustering (for example K-Means (MacQueen, 1967)), observations should be similar both in the descriptive and in the temporal space. We consider that each entity ϕ_l is associated at every moment of time with one and only one evolution phase, *i.e.*, each observation x_i belongs to a single evolution phase. Furthermore, phases are assumed to be linked to each other, to allow entities to temporally navigate between them. A temporal succession of evolution phases forms an *evolution path*. Therefore, the "slow changing world" hypothesis, which assumes that long-term trends evolve slowly, translates into evolution paths in which successive evolution phases have high connection strengths.

Prototypes. We define $\mu = (\mu^t, \mu^d), \mu \in \mathscr{M}$ the prototype of an evolution phase $\mathscr{C}$, where $\mathscr{M}$ is the set of all prototypes. Just like centroids in traditional clustering, prototypes behave likes "central tendencies" of their phases and characterize the phases both in the temporal and in the descriptive space. Unlike centroids, the prototypes cannot be rigorously defined outside the learning problem, as they are dependent not only on the observations



(a) (b)

Fig. 1: An example of a desired output, in which the evolutions of 3 entities (ϕ_1, ϕ_2 and ϕ_3) are described using 5 phases ($\mathscr{C}_i, i = 1,\ldots,5$). For example, the evolution path of ϕ_3 is $\mathscr{C}_1 \rightarrow \mathscr{C}_3 \rightarrow \mathscr{C}_3$. (a) The graph structure of the evolution phases. The arcs between two phases $(\mathscr{C}_i, \mathscr{C}_j)$ are labeled with the entities presenting the transition $\mathscr{C}_i \rightarrow \mathscr{C}_j$. (b) The observations of the 3 entities are partitioned contiguously into the 5 phases.

assigned with a phase, but also on the other prototypes and the choice of parameters (shown in Sect. 3.3).

Method. We infer typical evolution paths, by clustering the observations corresponding to entities into k clusters, which serve as *evolution phases*. The links between the evolution phases are represented using a graph structure, defined by its adjacency matrix $A = (a_{ij})$, where $a_{i,j} \in [0,1]$ is the strength of the link between clusters $\mathscr{C}_i$ and $\mathscr{C}_j$. A value of 0 denotes the absence of a link. The connection strength of the link is proportional to i) the similarity of their prototypes μ_i and μ_j, both in the temporal and descriptive space, and ii) the number of entities presenting the passage from $\mathscr{C}_i$ to $\mathscr{C}_j$. The graph is oriented and, therefore, the matrix A is not symmetrical. Fig. 1 shows the desired result of our clustering algorithm. Fig. 1a shows how the phases $\mathscr{C}_j$ are structured into a graph structure and, in Fig. 1b, the series of observations belonging to each entity are assigned to phases, thus forming continuous segments. The succession of segments is interpreted as the entity's evolution path.

We define the following objectives for the resulting partition:

- ***Obj. 1***: construct clusters which are coherent in the temporal and the descriptive space. Observations under a cluster should have similar descriptions (just as traditional clustering does) and they should be temporally close. Each cluster should provide a trade-off between the temporal and descriptive coherence, since the two might be contradictory. For example, two different periods with similar evolutions (*e.g.*, two economical crises) should be regrouped separately, as they represent two distinct evolution phases;
- ***Obj. 2***: segment, as contiguously as possible, the series of observations for each entity. The sequence of segments is interpreted as the entity's evolution path;
- ***Obj. 3***: present smooth passages between phases on evolution paths. An evolution path should take an entity through highly similar evolution phases, *i.e.*, changes should come in small increments.

3.2 Constructing the Objective Measure

ClusPath fulfills the aforementioned objectives by a) translating them into several objectives and combining them into an overall objective function $\mathscr{J}$ and b) applying a gradient descent optimization method, in which $\mathscr{J}$ is minimized, using a K-Means-like iterative relocation framework. To optimize ***Obj 1***, we use the temporal-aware dissimilarity measure proposed by Rizoiu et al (2012):

$$||x_i - x_j||_{TA} = 1 - \left(1 - \gamma_d \frac{||x_i^d - x_j^d||^2}{\Delta d_{max}^2}\right)\left(1 - \gamma_t \frac{||x_i^t - x_j^t||^2}{\Delta t_{max}^2}\right) \quad (1)$$

where γ_d and γ_t are controlled by the parameter $\alpha \in [-1,1]$, which acts like a slider, favoring the temporal component for $\alpha = -1$ or the descriptive component for $\alpha = 1$:

$$\gamma_d = \begin{cases} 1+\alpha, & \text{if } \alpha \in [-1,0] \\ 1, & \text{if } \alpha \in (0,1] \end{cases} \; ; \; \gamma_t = \begin{cases} 1, & \text{if } \alpha \in [-1,0] \\ 1-\alpha, & \text{if } \alpha \in (0,1] \end{cases} \quad , \tag{2}$$

$||\bullet||_{TA} \in [0,1]$ and a value of zero means identical observations. Δd_{max} and Δt_{max} are the diameters of the descriptive and temporal space respectively (the largest distance encountered between two observations). The temporal-aware measure allows to simultaneously take into account the similarity in the descriptive and temporal spaces, between two observations. By using this measure, ClusPath minimizes the term: $\sum_{\mu_p \in \mathscr{M}} \sum_{x_i \in \mathscr{C}_p} ||x_i - \mu_p||_{TA}$, where μ_p is the abstraction of cluster $\mathscr{C}_p$, to which the observation x_i is assigned.

The second objective (***Obj. 2***) states that, for comprehension reasons, the series of observations belonging to an entity should be segmented contiguously. We extend the initial segmentation mechanism of TDCK-Means, by setting *temporally-oriented must-link soft constraints* between each pair of observations, belonging to the same entity. Each entity is associated with N observations, therefore each observation is involved in $N-1$ constraints, linking it to all the other observations. Each constraint x_i has a direction, *i.e.*, it is temporally-oriented. $i-1$ of these constraints are incoming (from preceding observations to x_i), while $N-i$ are outgoing, towards the subsequent observations. A total of $(N-1)(N-2)/2$ constraints are set for each entity. A must-link constraint indicates that the two observations should be placed into the same cluster. Being soft constraints, ClusPath is allowed to break any number of them, while a time-decaying penalty is inflicted for each violation. The penalty is more severe when the observations are closer in time and less severe when the two assigned clusters have a strong link (high value for $a_{i,j}$). The penalty function is:

$$w(x_i,x_k) = \beta * e^{-\frac{1}{2}\left(\frac{||x_i^t - x_k^t||}{\delta}\right)^2} \left(1 - a_{j,l}^2\right) \mathbb{1}\left[x_i^\phi = x_k^\phi\right] \mathbb{1}\left[x_i^t < x_k^t\right] \quad , \tag{3}$$

where $x_i \in \mathscr{C}_j$, $x_k \in \mathscr{C}_l$ and $\beta \in \mathbb{R}^+$ is the weight of the penalty function; $\delta \in \mathbb{R}^+$ is a parameter which controls the width of the function. The penalty function in Eq. 3 is inspired from the Normal Distribution function and it does not require the discretization of time. Respecting all constraints involves associating all observations of an entity to the same evolution phase. Therefore, the segmentation mechanism strives to acquire a trade-off between clustering observations together and putting them into separate, yet well connected clusters. We obtain the first term (T_1) of the objective function, dealing with assigning observations to clusters:

$$T_1 = \sum_{\mu_p \in \mathscr{M}} \sum_{x_i \in \mathscr{C}_p} \left(||x_i - \mu_p||_{TA} + \sum_{\substack{x_k \in \mathscr{C}_q \\ q \neq p, x_i^\phi = x_k^\phi}}^{x_i^t < x_k^t} \beta * e^{-\frac{1}{2}\left(\frac{||x_i^t - x_k^t||}{\delta}\right)^2} \left(1 - a_{p,q}^2\right) \right) \quad . \tag{4}$$

The following terms of the objective function, leverage the influence of the graph structure into the objective function. In Sect. 3.1, we stated that the strength of the link between $\mathscr{C}_p$ to $\mathscr{C}_q$ is proportional to the similarity of their respective prototypes, μ_p and μ_q. We use the temporal-aware dissimilarity measure to assess the similarity of the prototypes. A

low value for $||\mu_p - \mu_q||_{TA}$ (high similarity) results in a high value for $a_{p,q}$ (powerful link between $\mathscr{C}_p$ and $\mathscr{C}_q$). This translates into the second term of the objective function:

$$T_2 = \sum_{\mu_p \in \mathscr{M}} \sum_{\substack{\mu_q \in \mathscr{M} \\ p \neq q}} a_{p,q}^2 ||\mu_p - \mu_q||_{TA} \ . \tag{5}$$

T_2, together with $\left(1 - a_{p,q}^2\right)$ in term T_1 (Eq. 4), assures the smooth passage for entities. T_1 encourages successive observations to be assigned to clusters with a high value for $a_{p,q}$. T_2 ensures that similar clusters have a strong link (high value for $a_{p,q}$). Therefore, successive observations belonging to an entity are assigned to similar clusters, satisfying the third learning objective (***Obj. 3***). The strength of the link between two clusters $\mathscr{C}_p$ and $\mathscr{C}_q$ is also dependent on the number of entities which present a transition from $\mathscr{C}_p$ to $\mathscr{C}_q$. We consider that an entity ϕ_l presents a transition between $\mathscr{C}_p$ to $\mathscr{C}_q$ ($\mathscr{C}_p \xrightarrow{\phi_l} \mathscr{C}_q$) if and only if two consecutive observations exist, associated with the given entity, where the first observation (ordered by their timestamp) is clustered under $\mathscr{C}_p$ and the second one under $\mathscr{C}_q$. We define the intersection similarity measure between two phases, based on the normalized number of entities that present the transition between the two phases:

$$inter_\phi(\mathscr{C}_p, \mathscr{C}_q) = 1 - \frac{|\{\phi_l \in \Phi | \mathscr{C}_p \xrightarrow{\phi_l} \mathscr{C}_q\}|}{|\Phi|} \ , \tag{6}$$

where $inter_\phi(\mathscr{C}_p, \mathscr{C}_q) \in [0,1]$ and a value of zero means that all entities present a transition from $\mathscr{C}_p$ to $\mathscr{C}_q$. We construct the third term of the objective function:

$$T_3 = \sum_{\mu_p \in \mathscr{M}} \sum_{\substack{\mu_q \in \mathscr{M} \\ p \neq q}} a_{p,q}^2 inter_\phi^2(\mathscr{C}_p, \mathscr{C}_q) \ . \tag{7}$$

We construct the objective function $\mathscr{J}$ as the weighted sum of the three terms in Eq. 4, 5 and 7:

$$\begin{aligned} \mathscr{J} &= \lambda_1 T_1 + \lambda_2 T_2 + \lambda_3 T_3 = \\ &= \lambda_1 \sum_{\mu_p \in \mathscr{M}} \sum_{x_i \in \mathscr{C}_p} \left(||x_i - \mu_p||_{TA} + \sum_{\substack{x_k \in \mathscr{C}_q \\ q \neq p,\, x_i^\phi = x_k^\phi}}^{x_i^t < x_k^t} \beta * e^{-\frac{1}{2}\left(\frac{||x_i^t - x_k^t||}{\delta}\right)^2} \left(1 - a_{p,q}^2\right) \right) \\ &+ \lambda_2 \sum_{\mu_p \in \mathscr{M}} \sum_{\substack{\mu_q \in \mathscr{M} \\ p \neq q}} a_{p,q}^2 ||\mu_p - \mu_q||_{TA} + \lambda_3 \sum_{\mu_p \in \mathscr{M}} \sum_{\substack{\mu_q \in \mathscr{M} \\ p \neq q}} a_{p,q}^2 inter_\phi^2(\mathscr{C}_p, \mathscr{C}_q) \ , \end{aligned} \tag{8}$$

where $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}^+$ are parameters of the algorithm and they represent the weights of the different components. They allow to fine-tune the impact of each of the stated objectives. For example, if it is desirable to obtain evolution paths with small increments, in which the successive phases are very similar, it suffices to set λ_2 to a high value. It would also result in paths with a larger number of phases than for a combination of weights featuring a lower λ_2.

The objective function $\mathscr{J}$ in Eq. 8 is artificially minimized if $a_{pq} = 0, \forall p, q$. Given that $\mathscr{J} \in \mathbb{R}^+$, we constrain the 1-norm of the adjacency matrix A:

$$||A||_1 = 1 \Leftrightarrow \sum_{p=1}^{k} \sum_{q=1}^{k} a_{p,q} = 1 \ . \tag{9}$$

This strategy allows to set high values of $a_{p,q}$ for pairs of clusters for which i) $||\mu_p - \mu_q||_{TA}$ is low (similar prototypes) and ii) $inter^2_\phi(\mathscr{C}_p, \mathscr{C}_q)$ is low (a high number of entities present a transition from $\mathscr{C}_p$ to $\mathscr{C}_q$).

3.3 Optimizing the Objective Function. The ClusPath Algorithm.

We minimize the objective function $\mathscr{J}$ using a K-Means-like algorithm, by estimating the inner variables: i) the assignment of observations to clusters, ii) the prototypes of clusters and iii) the adjacency matrix. ClusPath uses an iterative relocation strategy, at each step considering as fixed two of the variables (*e.g.*, the observations assignment and the prototypes) and analytically computing the values of the third (the adjacency matrix).

Assignment of observations to clusters. For each observation x_i, ClusPath chooses a cluster $\mathscr{C}_p$ so that $\mathscr{J}$ is minimized. Considering that T_2 and T_3 in Eq. 8 are not dependent on the assignment of observations to clusters, choosing a cluster for x_i boils down to minimizing T_1:

$$\textbf{best\_cluster}(x_i) = \underset{p=1,2,\ldots,k}{argmin} \left(||x_i - \mu_p||_{TA} + \sum_{\substack{x_k \in \mathscr{C}_q \\ q \neq p \\ x_i^\phi = x_k^\phi}}^{x_i^t < x_k^t} \beta * e^{-\frac{1}{2}\left(\frac{||x_i^t - x_k^t||}{\delta}\right)^2} \left(1 - a^2_{p,q}\right) \right) \tag{10}$$

This guaranties that the contribution of x_i to the value of $\mathscr{J}$ diminishes or stays constant. Overall, this assures that $\mathscr{J}$ does not increase in the assignment phase.

Recomputing the prototypes of the clusters. ClusPath updates the prototypes by recomputing one prototype at a time (*e.g.*, μ_j), while considering all the other prototypes fixed at their values from the previous iteration. Each of the k prototypes $\mu_j \in \mathscr{M}$ is composed from a descriptive component and a temporal component: $\mu_j = (\mu_j^d, \mu_j^t)$. Given that the subproblem of recomputing one prototype is quadratic and no additional constraints are imposed on μ_j, each of the two components is updated individually, by calculating the fixed point. Given that only the first two terms of the function $\mathscr{J}$ are dependent of μ_j, from Eq. 8 we obtain the prototype update formulas (details of the complete calculations are in the Supplementary Materials (SM) (sup, 2016)):

$$\begin{aligned}
\mu_j^d &= \frac{\lambda_1 \sum_{x_i \in \mathscr{C}_j} x_i^d \left(1 - \gamma_t \frac{||x_i^t - \mu_j^t||^2}{\Delta t^2_{max}}\right) + \lambda_2 \sum_{\substack{\mu_p \in \mathscr{M} \\ p \neq j}} \mu_p^d \left(1 - \gamma_t \frac{||\mu_p^t - \mu_j^t||^2}{\Delta t^2_{max}}\right)(a^2_{j,p} + a^2_{p,j})}{\lambda_1 \sum_{x_i \in \mathscr{C}_j} \left(1 - \gamma_t \frac{||x_i^t - \mu_j^t||^2}{\Delta t^2_{max}}\right) + \lambda_2 \sum_{\substack{\mu_p \in \mathscr{M} \\ p \neq j}} \left(1 - \gamma_t \frac{||\mu_p^t - \mu_j^t||^2}{\Delta t^2_{max}}\right)(a^2_{j,p} + a^2_{p,j})} \\
\mu_j^t &= \frac{\lambda_1 \sum_{x_i \in \mathscr{C}_j} x_i^t \left(1 - \gamma_d \frac{||x_i^d - \mu_j^d||^2}{\Delta d^2_{max}}\right) + \lambda_2 \sum_{\substack{\mu_p \in \mathscr{M} \\ p \neq j}} \mu_p^t \left(1 - \gamma_d \frac{||\mu_p^d - \mu_j^d||^2}{\Delta d^2_{max}}\right)(a^2_{j,p} + a^2_{p,j})}{\lambda_1 \sum_{x_i \in \mathscr{C}_j} \left(1 - \gamma_d \frac{||x_i^d - \mu_j^d||^2}{\Delta d^2_{max}}\right) + \lambda_2 \sum_{\substack{\mu_p \in \mathscr{M} \\ p \neq j}} \left(1 - \gamma_d \frac{||\mu_p^d - \mu_j^d||^2}{\Delta d^2_{max}}\right)(a^2_{j,p} + a^2_{p,j})}
\end{aligned} . \tag{11}$$

Similarly to K-Means, the prototypes computed in Eq. 11 represent prototypes of their clusters. Unlike K-Means, μ_j the prototype of a cluster $\mathscr{C}_j$ is not only the average of the observations regrouped under $\mathscr{C}_j$, but it is also influenced by the prototypes of the other clusters linked to $\mathscr{C}_j$. Moreover, it represents the average of the assigned observations on both the temporal dimension, as well as on the descriptive dimension, with the two dimensions

weighting each other. *E.g.*, temporally central observations weight more in the calculation of μ_j^d, the descriptive component of the new prototype. Furthermore, the contributions of the other prototypes (μ_p) are weighted by the strength of the link between $\mathscr{C}_p$ and $\mathscr{C}_j$, the cluster of the currently recomputed prototype μ_j. Inferring the relations between clusters simultaneously with the clustering represents one of the advantages of ClusPath: the relations between clusters can also influence their content. The orientation of the link is not important, since both $a_{p,j}$ and $a_{j,p}$ appear in the update formula.

Updating the adjacency matrix. Similarly to recomputing the prototypes, the update of the adjacency matrix is a quadratic problem. The difference is that the additional constraint in Eq. 9 is imposed, in order to avoid a trivial null solution. A typical solution to optimizing an equality constraint problem is to use a Lagrange multiplier on the constraint, similarly to (Dunn, 1973). Consequently, the update formula for the adjacency matrix is the solution of the following:

$$\frac{\partial \mathscr{J}^*}{\partial a_{r,s}} = 0, \text{ where } \mathscr{J}^* = \mathscr{J} - \lambda \left(\sum_{p=1}^{k} \sum_{q=1}^{k} a_{p,q} - 1 \right) \quad . \tag{12}$$

By computing the point in which the derivative of $\mathscr{J}^*$ is null, we obtain the adjacency matrix update formulas (for complete calculations see SM (sup, 2016)):

$$\begin{aligned} a_{r,s} = & \frac{1}{K_{r,s} \sum_{p=1}^{k} \sum_{q=1}^{k} \frac{1}{K_{p,q}}} \quad , \\ & \text{with } K_{r,s} = -\lambda_1 \; pen(\mathscr{C}_r \xrightarrow{\phi} \mathscr{C}_s) + \lambda_2 ||\mu_r - \mu_s||_{TA} + \lambda_3 \; inter_\phi^2(\mathscr{C}_p, \mathscr{C}_q) \quad , \\ & \text{and } pen(\mathscr{C}_r \xrightarrow{\phi} \mathscr{C}_s) = \sum_{x_i \in \mathscr{C}_r} \sum_{\substack{x_k \in \mathscr{C}_s \\ x_i^\phi = x_k^\phi}}^{x_i^t < x_k^t} \left(\beta * e^{-\frac{1}{2}\left(\frac{||x_i^t - x_k^t||}{\delta}\right)^2} \right) \quad . \end{aligned} \tag{13}$$

The $pen(\mathscr{C}_r \xrightarrow{\phi} \mathscr{C}_s)$ function in Eq. 13 is the influence that the contiguous penalty defined in Eq. 3 has on the adjacency update process. Intuitively, this mechanism allows to take into account the transitions of entities between phases, when computing the adjacency matrix. If many entities present transitions between clusters $\mathscr{C}_r$ and $\mathscr{C}_s$, then $a_{r,s}$ will be recomputed at a large value. This, in turn, allows in the subsequent iteration to lower the contiguity penalty, by lowering the $1 - a_{r,s}^2$ term in Eq. 3. The similarity between the prototypes μ_r and μ_s and the number of entities presenting a transition between $\mathscr{C}_r$ and $\mathscr{C}_s$ also impact the adjacency matrix update through $K_{r,s}$.

The ClusPath Algorithm. The outline of ClusPath is given in Algorithm 1. ClusPath seeks to minimize $\mathscr{J}$ by iterating an assignment phase, a prototype update phase and an adjacency matrix update phase until the partition does not change between two iterations. Aside from k (the number of clusters), ClusPath uses six parameters: α, β, δ, $\lambda_1, \lambda_2, \lambda_3$. In Sect. 3.4, we discuss a technique for tuning these parameters to an unseen dataset. A random subset of observations $x_l \in \mathscr{X}, 1 = 1, 2, ..k$ can be used as $\mathscr{M}^{(0)}$, the initial set of prototypes. The **up_cluster** and **up_adjacency** are two functions used to recompute, respectively, the prototypes and the adjacency matrix, as shown before in this section. Similarly to other gradient descent algorithms, ClusPath may present the usual shortcomings, such as converging to a local optima or slow convergence speeds near the minimum. While the convergence speed of ClusPath has not been theoretically studied, the experiments in Section 4 show that the optimization process practically stops in a few steps.

Algorithm 1: Outline of the ClusPath Algorithm.

Data: observations $x_i \in \mathscr{X}$, set of initial prototypes $\mathscr{M}^{(0)}$
Result: prototypes $\mu_j, j = 1,..,k$, clusters $\mathscr{C}_j, j = 1,..,k$, adjacency matrix A
Parameters: number of clusters k, α, β, δ, $\lambda_1, \lambda_2, \lambda_3$
// adjacency matrix initialization
$a_{i,j}^{(0)} = 0, \ \forall i, j = 1,2,..,k$
$iter \leftarrow 0$
$\mathscr{P}^{(iter)} \leftarrow \emptyset$ *//set of phases*
repeat
  $iter \leftarrow iter + 1$
  for $j = 1,2,...,k$ **do**
    $\mathscr{C}_j^{(iter)} \leftarrow \emptyset$
  // S1. observation assignment to phases
  for $x_i \in \mathscr{X}$ **do**
    $j =$ **best_cluster**(x_i, $\mathscr{X}$, $\mathscr{M}^{(iter-1)}$, $\mathscr{P}^{(iter-1)}$, $A^{(iter-1)}$)
    $\mathscr{C}_j^{(iter)} = \mathscr{C}_j^{(iter)} \cup \{x_i\}$
  // S2. update prototypes
  for $j = 1,2,...,k$ **do**
    $(\mu_j^{d,(iter)}, \mu_j^{t,(iter)}) \leftarrow$ **up_prototype**(j, $\mathscr{X}$, $\mathscr{M}^{(iter-1)}$, $\mathscr{P}^{(iter-1)}$, $A^{(iter-1)}$)
  // S3. update adjacency matrix
  $A^{(iter)} \leftarrow$ **up_adjacency**($\mathscr{X}$, $\mathscr{M}^{(iter-1)}$, $\mathscr{P}^{(iter-1)}$)
  $\mathscr{M}^{(iter)} \leftarrow \{\mu_j^{(iter)} | \ j = 1,2,..,k\}$
  $\mathscr{P}^{(iter)} \leftarrow \{\mathscr{C}_j^{(iter)} | \ j = 1,2,..,k\}$
until $\mathscr{C}_j^{(iter)} = \mathscr{C}_j^{(iter-1)}, \forall j = 1,2,..,k$

Algorithm complexity. We denote by $T(x)$ the time complexity of subroutine x. From Algorithm 1 is results $T(\text{ClusPath}) = T(S1) + T(S2) + T(S3)$. If p is the number of entities and N is the number of observations associated with each entity, then $n = p \times N$ is the total number of observations. We assume that $k << n$. We compute $T(S1)$ as $nT(\textbf{best\_cluster}) = pNT(\textbf{best\_cluster})$. Due to the penalty term in Eq. 10, $T(\textbf{best\_cluster}) = \mathscr{O}(kN)$, therefore $T(S1) = \mathscr{O}(pN^2k)$. The complexity of updating centroids ($S2$) is $kT(\textbf{up\_prototype})$. From Eq. 11 results $T(\textbf{up\_prototype}) = 2(\mathscr{O}(n) + \mathscr{O}(k)) = \mathscr{O}(n)$, therefore $T(S2) = \mathscr{O}(pNk))$. Lastly, $T(S3) = T(\textbf{up\_adjacency}) = k^2T(K_{r,s})$, which can be obtained from Eq. 13. Computing $a_{r,s}$ is dependent on computing $K_{r,s}$, which need be computed only once per iteration. $T(K_{r,s}) = T\left(inter_\phi^2(\mathscr{C}_p, \mathscr{C}_q)\right) + T\left(pen(\mathscr{C}_r \xrightarrow{\phi} \mathscr{C}_s)\right)$. From Eq. 6 results $T\left(inter_\phi^2(\mathscr{C}_p, \mathscr{C}_q)\right) = \mathscr{O}(pN)$ (since we need to iterate only once through the observations of an entity to detect transitions). Furthermore, $T\left(pen(\mathscr{C}_r \xrightarrow{\phi} \mathscr{C}_s)\right) = \mathscr{O}(nN) = \mathscr{O}(pN^2)$. Consequently, $T(K_{r,s}) = \mathscr{O}(pN^2) \Rightarrow T(S3) = \mathscr{O}(pN^2k^2)$. This amounts to a complexity of ClusPath of $\mathscr{O}(pN^2k^2)$, which is well adapted to Social Science and Humanities datasets, where often a large number of entities is studied over a relatively short period of time ($p > N$).

Heuristics for Displaying the Constructed Graph. The adjacency matrix $A = (a_{i,j})$ shows the strength of the link between each pair of clusters. $a_{i,j} \in \mathbb{R}, i, j = 1,2,..,k$. To display the relation between clusters as a graph, we construct a binary matrix A^*, using a simple heuristic: a threshold λ is chosen so that only the $k-1$ scores are retained. This value is chosen to favor a tree structure, even if a tree cannot be guaranteed given the structure of the graph (*i.e.*, some nodes may be central, with many connections, while others are marginal or even unconnected). All arcs having the selected scores are plotted, consequently, more than $k-1$ arcs might be used. $a_{i,j}^* = 1$ *iff* $a_{i,j} > \lambda$, and $a_{i,j}^* = 0$ *iff* $a_{i,j} \leq \lambda$. Unconnected

nodes are eliminated, as they are considered isolated evolution phases and, therefore, not suitable for an evolution path. Given the adjacency matrix update formula in Eq. 13, $a_{i,j}$ is low when entities do not present transitions between $\mathscr{C}_i$ and $\mathscr{C}_j$ and μ_i and μ_j are very dissimilar. Therefore, entities which find themselves in these unconnected phases do not transition into other phases and are dissimilar to the others. In other words, they act as outliers of the typical evolution path.

3.4 Automatically Tuning the Parameters Using Evolutionary Algorithms

ClusPath uses six parameters: α, β, δ, $\lambda_1, \lambda_2, \lambda_3$, which can prove challenging to tune, especially on new, unseen datasets. Datasets issued from different domains, like the ones presented in Sect. 4, may have different requirements for the ratio between the descriptive and temporal dimensions (the α parameter) or the smoothness of the evolution path (the λ_2 parameter). We provide an optional method for automatically tunning the parameters of ClusPath, at the expense of repeated runs of the algorithm. Using an evolutionary technique, we optimize over the 6-dimensional space of the parameters in order to find a solution which provides a balance between the four opposite measures used to evaluate its output.

Evaluating a partition. We use the classical Information Theory measures to numerically assess the four objectives defined in Sect. 3.1. The first three objectives are evaluated using measures proposed by Rizoiu et al (2012). The coherence of the obtained partition in the descriptive and temporal dimensions are measured using the classical variance on, respectively, the multidimensional component (*MDvar* measure) and the temporal component (*Tvar* measure). We compute the variance as the mean within-cluster dissimilarity between observations and their associated prototype. The contiguous segmentation of the series of observations corresponding to an entity is measured using a penalized Shannon entropy (*ShaP*), in which a sequentiality component is added by weighting the value of the classical entropy by a penalty factor depending on the number of continuous segments in the series of each entity:

$$ShaP = \frac{1}{|\Phi|} \sum_{\phi \in \Phi} \sum_{i=1}^{k} \left[-p_\phi(\mathscr{C}_i) \log_2(p_\phi(\mathscr{C}_i)) \right] \left(1 + \frac{n_{ch} - n_{min}}{N-1} \right), \quad p_\phi(\mathscr{C}_i) = \sum_{\substack{x_j \in \mathscr{C}_i \\ x_j^\phi = \phi}} \frac{1}{N}$$

where n_{ch} is the number of changes in the cluster assignment series of an entity, n_{min} is the minimal required number of changes and N is the number of observations for an entity. We add a forth measure (*SPass*) to assess the smooth passage of entities along an evolution path measure:

$$SPass = \sum_{\phi \in \Phi} \sum_{\mathscr{C}_i \xrightarrow{\phi} \mathscr{C}_j}^{i,j \in 1,\ldots,k} \frac{||\mu_i - \mu_j||_{TA}}{n_{ch}} \ .$$

which measures the average temporal-aware dissimilarity between successive phases. Therefore, to each solution constructed by ClusPath corresponds a point in the four-dimensional space of the measures: (*MDvar*, *Tvar*, *ShaP*, *SPass*)

Defining a balanced solution. The parameter tuning heuristic. The four objectives defined in Sect. 3.1 and the four corresponding measures defined here above are contradicting; completely fulfilling them at the same time is not possible. To acquire a trade-off between the mutually contradicting objectives, we pose the problem of automatically tuning the parameters of ClusPath as an optimization problem. We consider the parameters of ClusPath as internal variables over which the search is performed and we evaluate using the four evaluation criteria. To chose a balanced solution, we use the concept of Pareto optimality (Sawaragi et al, 1985), originally developed in economics. Given a set of solutions, a

given solution is considered to be Pareto optimal if there exists no other that, simultaneously, is better on all objectives. The set of Pareto optimal solutions form the Pareto front. Consequently, no single optimum can be constructed, but rather a class of optima, depending on the ratio between the objectives. With no *a priori* information, selecting the point on the Pareto front closest (in terms of Euclidean distance) to the ideal point provides a good compromise between the different objectives. All measures need to be minimized, therefore the ideal point is $(0,0,0,0)$. By default, we use no weights on the dimensions when calculating the distance from a solution to this point. Before computing the Euclidean distance to the ideal point, the values of all measures are normalized, in order to receive equal importance. Having the Pareto constructed allows using another methodologies for selecting the "good trade-off" without re-executing the lengthy optimization procedure.

Approximating the Pareto frontier using evolutionary algorithms. The heuristic we propose is, for a given dataset, to construct the Pareto front in the measures space, which is the 4-dimensional envelope of all the possible compromises. The solution closest to the ideal point is chosen as the "best" compromise and its corresponding parameter values are presented as the tuned values for the particular dataset. We formulate the problem of parameter tunning as a multi-objective optimization problem, optimizing in the 4-dimensional space of evaluation measures, with the parameters of ClusPath serving as internal variables over which the search is performed. Solving multiobjective optimization problems using evolutionary algorithms (MOEAs) has been investigated by many authors (Deb et al, 2002; Halsall-Whitney and Thibault, 2006; Kafafy et al, 2011; Mihăiţă et al, 2014; Zitzler et al, 2001). Pareto dominance based MOEAs such as NSGAII (Deb et al, 2002), SPEA2 (Zitzler et al, 2001) and HEMH (Kafafy et al, 2011) have been dominantly used in the recent studies. In multiobjective optimization, the set of Pareto optimal solutions is approximated using a large number of non-dominated points. MOEAs operates on individuals of an initial population to generate the individuals of the population of the next generation. The new population is generated by applying some processes of selection, recombination and mutation.

Our technique The genome of each individual is a vector composed of the six parameters of ClusPath: α, β, δ, $\lambda_1, \lambda_2, \lambda_3$. For each individual, ClusPath is executed with the parameters in its genome, a solution is obtained and it is evaluated. Therefore, to each individual corresponds a point in the 4-dimensional space of the measures. The size of each evolutionary population is fixed to 100. We use the Pareto dominance to evaluate the fitness function: the number of individuals which dominate the given individual. An elitist selection is used to filter the population: all the non-dominated solutions and 10% of the dominated solutions are promoted in the next generation, while the rest are removed. We use two operators to construct, based on the selected elite, new solutions until the population reaches the nominal size. We duplicate and mutate 5% of the survivors. The mutations affect one or two parameters in the genome, which are set to a random value in their domain of definition. New offsprings are generated through a *Path-Relinking* strategy: two parents are selected from the survivors and the values for the newly generated individuals are averaged means of the values of the parents. The weights are randomly generated. The process is iterated until all the constructed solutions are Pareto non-dominated or until a maximum number of generations (100) is reached. We choose as the "best" solution, the individual in the last generation which is the closest to the ideal point. Given the elitist strategy, non-dominated solutions always survive into successive generations.

The complexity of the evolutionary heuristic. For a given partition constructed by ClusPath, we have $T(MDvar) = T(Tvar) = T(ShaP) = \mathcal{O}(pNk)$ and $T(SPass) = \mathcal{O}(pN)$. Given m the number of individuals in each evolutionary generation, evaluating all the individuals in a generation takes $\mathcal{O}(mpNk)$. Calculating the fitness function at each generation

has the complexity $\mathcal{O}(m^2)$. Sorting the individuals by their fitness is done in $\mathcal{O}(m \times log(m))$. ClusPath is executed for each individual, therefore a complexity of $m \times \mathcal{O}(pN^2k^2) = \mathcal{O}(mpN^2k^2)$ (cf. complexity of ClusPath calculated in Sect. 3.3). Therefore, the complexity of each evolutionary generation is $\mathcal{O}(mpNk)+\mathcal{O}(m^2) + \mathcal{O}(mpN^2k^2) = \mathcal{O}(mpN^2k^2)$, considering that $m << pN^2k^2$. This results in a total complexity of the evolutionary heuristic of $\mathcal{O}(ng_{max} \times mpN^2k^2)$, where ng_{max} is the maximum number of generations.

Note that the calculated complexity is the worst case scenario. In practice, the elitist technique speeds up the computation considerably, since the solutions promoted from the previous generation do not require a re-execution of the ClusPath algorithm. Furthermore, the elitist technique speeds up convergence and reduces the actual number of required generations. Finally, evolutionary algorithms are very well adapted for massive parallelization. Each individual execution of ClusPath is independent of the others and all the individuals in a generation can be computed in parallel, provided that the heuristic is ran on a machine with at least m execution cores. In our experiments in Section 4, the evolutionary algorithm proved to be only 6.12 times slower than ClusPath (standard deviation of $0,46$ over 200 runs, all runtimes and hardware specs in SM (sup, 2016)).

4 Experiments

The experiments are conducted on two real-life datasets, one issued from political sciences: *Comparative Political Data Set I* (`CPDS1`) (Armingeon, Klaus, Christian Isler, Laura Knöpfel and Engler, 2011) and the second containing financial and accounting data: *European Companies* (`EC`) (Siddiqui et al, 2012). `CPDS1` is a collection of political and institutional data, which consists of annual data for 23 democratic countries for the period from 1960 to 2009. The dataset was cleaned by removing redundant variables (*e.g.* country identifier and postal code), resulting in each country being described using 207 political, demographic, social and economic variables. The corpus was preprocessed by removing entity-specific, time-invariant bias from the data. For every attribute, we compute its mean for each entity. For every pair (attribute, observation) we substract from the attribute value the mean of the corresponding entity. The obtained dataset[1] is under the form of triples $(country, year, description)$. The `EC` dataset describes the activity of 836 companies over a period of 5 years (2003-2007), using 7 economic variables. The dataset is preprocessed similarly to `CPDS1`, by removing the entity specific means from each variable. Note that the `EC` dataset is very different from `CPDS1`, in the sense that a consistently larger number of entities are studied over a short time span (5 timepoints for `EC` vs. 50 for `CPDS1`) and are described using few attributes (7 for `EC` vs. 207 for `CPDS1`). As the experiments in the following sections show, this leads to less diverse evolution phases and shorter evolution paths. All attributes are normalized prior to the execution of the algorithms (to avoid setting artificial weights), whatsoever the results presented in this section are non-normalized, in order to appreciate the amplitude of the time-varying component.

4.1 Choosing the Best Possible Solution Using an Evolutionary Algorithm

Throughout our experiments, the parameters of ClusPath (except for k, the number of clusters) are chosen using the evolutionary heuristic described in Sect. 3.4. Being a clustering algorithm, ClusPath suffers from the classical drawback of selecting the number of evolution phases. Setting k to large values results in clusters containing less observations, more compact in description and time, adapted for detecting granular evolutions. It also result in longer evolution paths. Setting k to a low number results in more general evolution phases, more

[1] Download pretreated version of the `CPDS1` dataset here: `http://goo.gl/17ihsf`

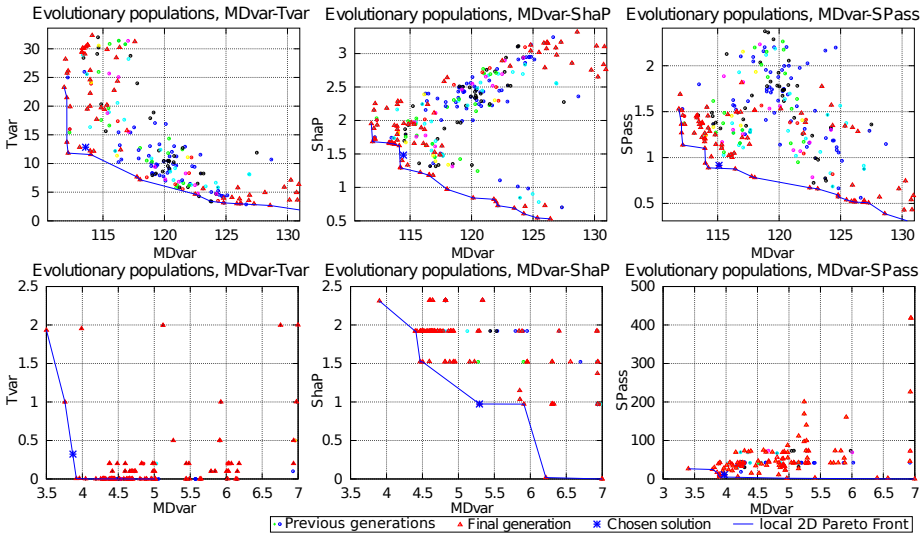


Fig. 2: Typical example of execution of the evolutionary algorithm on `CPDS1` (top row) and on `EC` (bottom row). The obtained 4-dimensional Pareto front is projected onto the (*MDvar*, *Tvar*) space (left), (*MDvar*, *ShaP*) space (middle) and (*MDvar*, *SPass*) space (right).

adapted to detecting more general evolutions. Fig. 2 presents a typical execution of the evolutionary algorithm, together with the obtained Pareto front and the chosen solution. Each individual in each evolutionary generation is associated with a point in the 4-dimensional space of the measures: (*MDvar*, *Tvar*, *ShaP*, *SPass*). For presentation reasons, we project the 4-dimensional space onto 2-dimensional spaces, by selecting pairs of measures. In Fig. 2, some of the points on the 4-dimensional Pareto front (indicated by the red triangles) seem Pareto dominated in the 2-dimensional spaces. The 2-dimensional graphics present only projections and, as discussed in Sect. 3.4, optimizing multiple criteria means finding a compromise which rarely yields the best results on the individual criteria. Similarly, the chosen global solution is not necessarily the optimum in each of the 2-dimensional spaces. Whatsoever, Fig. 2 clearly shows that the chosen point is never too far from the local Pareto fronts and, thus, it provides a good trade-off. On `EC`, the *Tvar* measure (bottom row, left and center graphics) presents levels. This is due to the very limited temporal extent of the dataset (5 years, *i.e.*, 5 data points), which in turn limits the number of values that can be taken by the temporal variance.

Runtime An average run of ClusPath on `CPDS1` takes 122.81s ($stdev = 11.37$) on our 24 cores Intel(R) Xeon(R) E5-2430 machine (see SM (sup, 2016) for full machine specs), while the evolutionary technique takes 750.69s ($stdev = 53.8$). This makes an execution of the evolutionary technique as long as roughly six sequential executions of ClusPath.

Qualitative Results Fig. 3 shows the typical evolution paths constructed by ClusPath on `CPDS1`, when asked for 20 clusters. The heuristic used for displaying the constructed graph eliminates unconnected phases. Fig. 3a shows how many countries belong in a certain cluster for each year. Clusters $\mathscr{C}_7$ (black), $\mathscr{C}_{11}$ (magenta) and $\mathscr{C}_{15}$ (blue) contain most of the observations, suggesting that the path $\mathscr{C}_7 \longrightarrow \mathscr{C}_{11} \longrightarrow \mathscr{C}_{15}$ is a typical evolution path followed by most entities. The meaning of each constructed cluster unravels when studying the segmentation of countries over clusters, in Fig. 3b, as well as the proposed graph

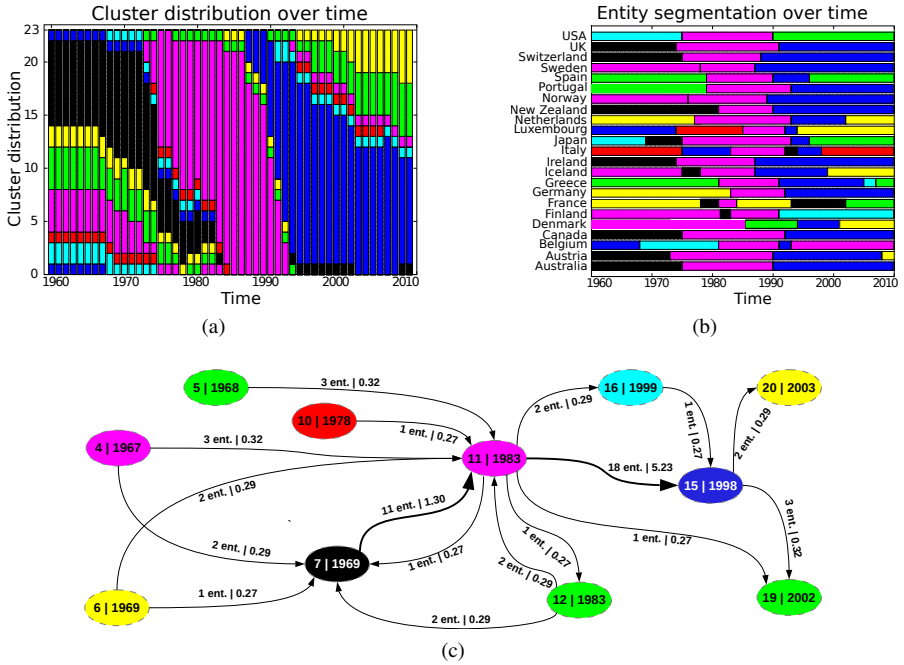
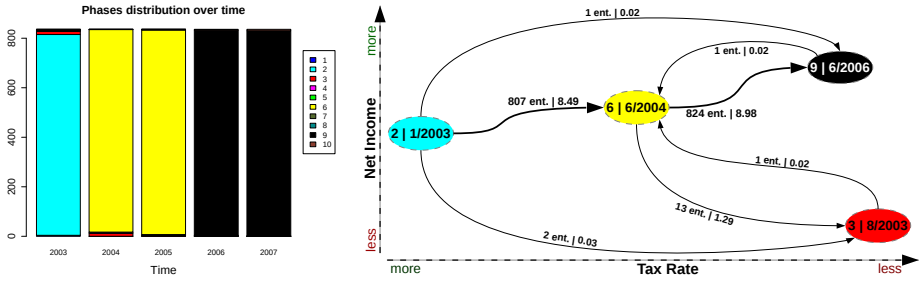


Fig. 3: Typical evolution phases constructed by ClusPath on CPDS1, with 20 clusters. Number of entities in each phase per year (a), segmentation of entities over phases (b) and the phase evolution graph (c)

structure in Fig. 3c. For example, the succession $\mathscr{C}_5 \longrightarrow \mathscr{C}_{11}$ is followed by Spain, Portugal and Greece at the beginning of their evolution. Historically, this coincides with the non-democratic regimes in those countries (Franco's dictatorship in Spain, the "Regime of the Colonels" in Greece). Likewise, the succession $\mathscr{C}_4 \longrightarrow \mathscr{C}_{11}$ (and the slightly longer $\mathscr{C}_4 \longrightarrow \mathscr{C}_7 \longrightarrow \mathscr{C}_{11}$) is present for countries like Denmark, Finland, Iceland, Norway and Sweden. This evolution path maps onto the "Swedish Social and Economical Model" of the Nordic countries. For completeness reasons, we present in Fig. 7, the graph of phases constructed *a posteriori* by the extensions of TDCK-Means, proposed in (Rizoiu et al, 2014). Visibly, the constructed paths are longer (most transitions are followed by a single entity) and more difficult to interpret. Furthermore, the entities evolution through the graph is not constrained during the clustering, which results in a significant number of *backwards links* from phases with a higher timestamp to phases with a lower timestamp. Overall, the graph generated by ClusPath is more synthetic and easier to interpret, since it contains less specific arcs (followed by a single entity) and less backward links. This makes is more adapted for our application: identifying typical evolution paths.

On the EC dataset, ClusPath is executed with 10 clusters, because of the considerably shorter temporal extent of the dataset. Figure 4a shows that the phases $\mathscr{C}_2$, $\mathscr{C}_3$, $\mathscr{C}_6$ and $\mathscr{C}_9$ are the predominant evolution phases, with the evolution $\mathscr{C}_2 \longrightarrow \mathscr{C}_6 \longrightarrow \mathscr{C}_9$ being the typical evolution in the population of companies. The descriptions of these phases are given in Table 1. Considering that the dataset was preprocessed to remove entity-specific values, the negative and positive values in Table 1 indicate negative, respectively positive, tendencies.



(a) (b)

Fig. 4: Typical evolution phases constructed by ClusPath on EC, with 10 clusters. Number of entities in each phase per year (a) the evolution graph projected in the space *NetIncome*/*TaxRate* (b)

For example, phase $\mathscr{C}_3$ is a crisis phase, in which companies reduce slightly their debt, while considerably reducing their revenues and income.

Figure 4b shows the obtained evolution graph, projected onto the 2-dimensional space defined by *NetIncome* and *TaxRate*, two of the descriptive variables in the dataset EC. The series of the *NetIncome* on the $\mathscr{C}_2 \longrightarrow \mathscr{C}_6 \longrightarrow \mathscr{C}_9$ evolution path is $-0.09 \longrightarrow -0.04 \longrightarrow 0.15$, whereas the *TaxRate* on the same evolution path is $0.08 \longrightarrow -0.04 \longrightarrow -0.06$. It depicts the "tax optimization" undertook by most companies in the dataset: most companies arrive to decrease significantly their tax rate, while increasing the net income. Phase $\mathscr{C}_3$ is a crisis phase, out of the 15 companies that enter it and only one exits it. This seems to indicate that in the economical climate preceding the crisis of 2009, one of the ways to keep a company profitable was fiscal optimization.

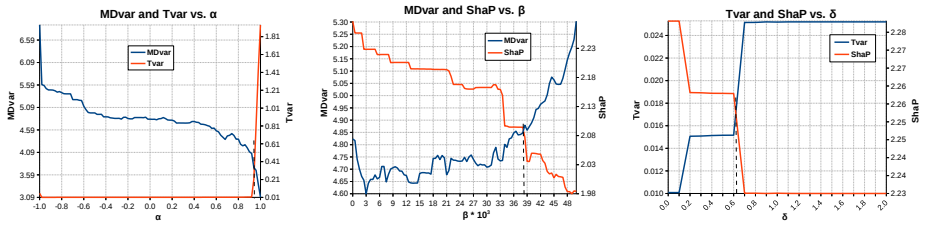
4.2 Quantitative Results

To handle the initialization bias present in clustering, we have constructed 20 sets of initial prototypes. Each of the tested algorithms was initialized identically, with each of the sets of initial prototypes and only the average values of the measures are reported. The performances of six algorithms are compared:

- **Simple K-Means** MacQueen (1967) clusters the observations based solely on their resemblance in the multidimensional space. Optimizes *MDvar*;
- **Temporal-Driven K-Means** Rizoiu et al (2012) uses K-Means with the the temporal-aware measure. Optimizes *MDvar* and *Tvar*. Parameters: $\alpha = 0$ and $\beta = 0$;

Table 1: Most common evolution phases in EC, described over the 7 dimensions of the dataset. The evolution path $\mathscr{C}_2 \longrightarrow \mathscr{C}_6 \longrightarrow \mathscr{C}_9$ is the path followed by most companies

Ph.	Time	FCFF	TotalDebt	Revenues	NetCapExp	EBITDA	TaxRate	NetIncome
$\mathscr{C}_2$	01/2003	-0.00	-0.01	-0.02	-0.00	-0.04	0.08	-0.09
$\mathscr{C}_3$	08/2003	-0.94	-0.06	-1.82	-0.67	-2.02	-0.07	-4.04
$\mathscr{C}_6$	06/2004	-0.01	-0.01	-0.02	-0.04	-0.02	-0.04	-0.04
$\mathscr{C}_9$	06/2006	0.05	0.01	0.07	0.04	0.07	-0.06	0.15



(a) (b) (c)

Fig. 5: Determine, using the heuristics described in the original paper, the values of TDCK-Means' parameters for dataset EC: α (a), β (b) and δ (c)

- **Constrained K-Means** Rizoiu et al (2012) uses the Euclidean distance and a penalty. Optimizes *MDvar* and *ShaP*. Parameters: $\alpha = 1$, $\beta = 0.0005$ and $\delta = 3$;
- **tcK-Means** Lin and Hauptmann (2006) is a temporal constrained clustering algorithm. It uses a threshold penalty function, adapted to the multi-entity case. Optimizes *MDvar* and *ShaP*. Parameters: $\alpha^* = 2, d^* = 4$.
- **TDCK-Means** Rizoiu et al (2012) uses the temporal-aware dissimilarity measure, as well as contiguity constraints. Optimizes *MDvar*, *Tvar* and *ShaP*. Parameters: $\alpha = 0.95$, $\beta = 0.0002$ and $\delta = 3$;
- **ClusPath** is the algorithm we propose in Sect. 3. Unlike the aforementioned algorithms, ClusPath is the sole algorithm to infer a graph structure for the clusters during the clustering. Optimizes *MDvar*, *Tvar*, *ShaP* and *SPass*. Parameters: determined automatically using the heuristic in Sect. 3.4.

Obtained results. The parameters of all algorithms, except ClusPath, are determined as shown in their original articles. For example, in Figure 5 we reproduce the heuristic of determining TDCK-Means' parameters on EC. Table 2 shows the average values of the measures, as well as the standard deviation (in italic) obtained by each algorithm. The best results on each measure are indicated in boldface. Note that, while ClusPath is designed to provide a compromise between the learning objectives, Simple K-Means, Temporal-Driven K-Means and Constrained K-Means are designed to optimize mainly one component. Not surprisingly, they show the best scores for, respectively, *MDvar*, *Tvar* and *ShaP*. ClusPath

Table 2: Mean value and standard deviation of evaluation measures for the different algorithms. All measures need to be minimized (best results in bold)

	Algorithm	*MDvar*		*Tvar*		*ShaP*		*SPass*	
CPDS1	K-Means	**114.89**	*3.8*	46.08	*8.03*	1.80	*0.18*	3.19	*0.63*
	Temporal Driven K-Means	125.32	*3.4*	**4.56**	*0.35*	2.96	*0.07*	1.52	*0.31*
	Constrained K-Means	140.26	*17.51*	163.60	*31.19*	**0.51**	*0.34*	1.13	*0.49*
	tcK-Means	131.54	*10.8*	156.15	*19.23*	0.61	*0.20*	2.15	*0.51*
	TDCK-Means	121.27	*4.34*	38.58	*7.16*	1.60	*0.13*	1.52	*0.55*
	ClusPath	118.68	*5.18*	6.26	*3.21*	2.81	*0.31*	**0.86**	*0.23*
EC	K-Means	**3.06**	*0.28*	1.95	*0.01*	0.15	*0.15*	8.41	*1.50*
	Temporal Driven K-Means	4.83	*0.43*	**0.01**	*0.03*	2.28	*0.12*	45.99	*9.63*
	Constrained K-Means	4.96	*0.69*	1.99	*0.01*	0.21	*0.03*	4.49	*6.56*
	tcK-Means	4.29	*0.34*	1.98	*0.01*	**0.04**	*0.01*	20.09	*14.61*
	TDCK-Means	4.41	*0.55*	0.07	*0.06*	2.14	*0.17*	5.03	*1.29*
	ClusPath	3.85	*0.59*	0.60	*0.25*	0.97	*0.22*	**4.40**	*1.25*

shows the best *SPass* score, proving that constructing the structure between clusters during the clustering process results in evolution paths with smoother passages. Both TDCK-Means and ClusPath seek to provide a trade-off between the measures, but ClusPath consistently outperforms TDCK-Means, except for the temporal variance on EC. Overall, ClusPath succeeds in providing a good trade-off between the different contradicting measures, obtaining the best *SPass* value and limited loss on the other measures.

4.3 Impact of Parameters and Result Stability

We launch the evolutionary heuristic on CPDS1 100 times with the same parameters of the evolutionary algorithm and with the same initial prototypes for each execution of ClusPath. This allows to i) assess the correlations between the chosen parameters and the obtained measures and ii) study the stability of the chosen solution, while lowering th impact of initialization randomness present in K-Means-like algorithms.

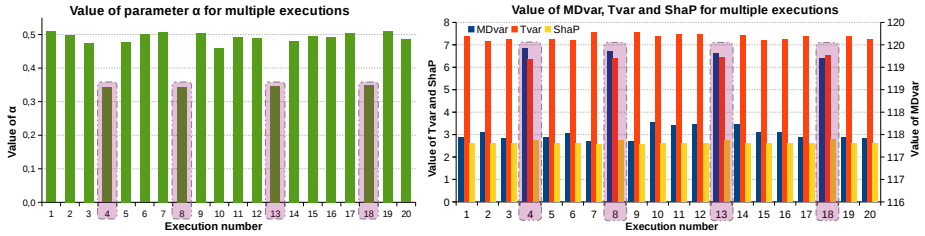
Impact of parameters on the evaluation measures Table 3 shows the Pearson correlation between i) the parameters and the measure, ii) between the parameters and iii) between the measures. Statistically significant correlations (with $p = 0.05$) are shown in boldface. The table on the right shows that the evaluation measures are, by pairs of two, correlated among themselves. The pairs a) *MDvar* and *ShaP* and b) *Tvar* and *SPass* are correlated positively. Conversely, *MDvar* and *ShaP* seem to be negatively correlated with Tvar and *SPass*. The result is consistent with the multiobjective optimization task: all solutions are scattered closely around the optimum solution and improving a measure mechanically involves degrading others, hence the negative correlations. The parameters with the most impact in ClusPath are α and β. They also have a weak correlation among themselves. α is statistically significantly correlated, negatively with *MDvar* and *ShaP*, and positively with *Tvar*. This was expected, considering that α is a slider variable, giving priority to the temporal component for higher values. β is negatively correlated with *MDvar* and *SPass* and positively correlated with *ShaP*: the higher β, the higher the contiguity penalty, which results in a more contiguous segmentation of observations for each entity.

Stability of the chosen solution for ClusPath We assess how stable are the solutions constructed by the evolutionary technique by studying the variability of the obtained parameters of ClusPath and the the values of evaluation measures. For parameter α, for most of the 100 execution described in this subsection, its values are distributed uniformly around 0.48. The exception are four cases in which α takes values around 0.35. These four "outlier" values of α, together with another 16 samples "regular" values (only 16 for readability purposes) are shown in Fig. 6a. There are no intermediary values between the two levels, which indicates that the Pareto front has two regions close to the ideal point: a larger one defined by values of α around 0.48 and a second, considerably smaller one, defined by $\alpha \approx 0.35$ (highlighted in Fig. 6a). Fig. 6b highlights the corresponding values of *MDvar*, *Tvar* and

Table 3: Correlation matrix i) between parameters and quality measures and ii) between parameters (left table) and iii) between quality measures (right table).

	MDvar	*Tvar*	*ShaP*	*SPass*	α	β	δ	λ_1	λ_2	λ_3
α	**-0.92**	**0.72**	**-0.90**	0.14	**1.00**	0.34	0.15	-0.07	-0.08	-0.08
β	**-0.31**	0.09	**0.30**	**-0.21**		**1.00**	-0.07	-0.20	-0.12	-0.06
δ	-0.17	0.05	-0.15	0.14			**1.00**	-0.08	0.07	-0.11
λ_1	0.05	-0.06	0.08	0.13				**1.00**	0.09	0.03
λ_2	0.11	-0.01	0.09	-0.09					**1.00**	0.00
λ_3	0.03	-0.06	0.03	0.07						**1.00**

	MDvar	*Tvar*	*ShaP*	*SPass*
MDvar	**1.00**	**-0.68**	**0.95**	**-0.31**
Tvar		**1.00**	**-0.76**	**0.34**
ShaP			**1.00**	**-0.37**
SPass				**1.00**



(a) (b)

Fig. 6: Two local optima: the region identified by negative values of α is highlighted (a) and the corresponding values for *MDvar*, *Tvar* and *ShaP* (b).

Table 4: Result stability: averages and coefficients of variation for all solutions (rows with *) and after removing the local optimum corresponding to lower α values (rows with **)

		α	$\beta \times 10^5$	δ	λ_1	λ_2	λ_3	MDvar	Tvar	ShaP	SPass
*	Average	0.477	7.727	3.02	484.74	536.51	522.35	117.69	7.33	2.61	0.76
	Coef. var.	7.11%	54.48%	18.35%	55.97%	52.76%	56.34%	0.32%	3.12%	1.35%	1.83%
**	Average	0.480	7.5	2.99	587.54	535.91	502.65	117.65	7.39	2.61	0.77
	Coef. var.	4.28%	59.63%	16.24%	37.22%	34.34%	61.48%	**0.17%**	**2.07%**	**0.85%**	**0.91%**

ShaP. This local minimum region of the Pareto front presents consistently elevated values of *MDvar*, slightly higher values of *ShaP* and lower values of *Tvar*. These observations are consistent with the conclusions of the previous paragraph, considering that α is correlated, i) negatively with *MDvar* and *ShaP* and ii) positively with *Tvar*. The existence of two local optima is confirmed by Table 4, which shows the mean and the coefficient of variation for the six parameters and four measures of ClusPath over all the executions (denoted by *) and after removing the four solutions corresponding to the local minima (denoted by **). The coefficients of variation of the evaluation measures for case * consistently decrease in the ** case. This proves that the removed solutions were caught in a local optimum region and the remaining solutions are grouped even more densely in the 4-dimensional space of the measures.

5 Conclusion and Future Work

In this paper, we have studied the construction of typical evolution paths followed by a collection of entities. We have proposed a novel algorithm, ClusPath, that partitions the observations belonging to entities into clusters, coherent in both the descriptive and temporal spaces. The connexions between clusters are inferred during the clustering process and successions of linked clusters are interpreted as evolution paths. A semi-supervised technique is used to leverage the strength of the links between clusters in the assignment of observations. An evolutionary technique is used to find the set of optimum parameters and choose the "best" trade-off of measures. We perform experiments on two real-live datasets, one issued from political sciences and the other issued from economics. We have shown how complex notions, such as socio-economical models (*i.e.*, the "Swedish" model) or tax policies (*i.e.*, the tax optimization performed by companies) can be detected from the temporal evolution of descriptive features.

The main novelty of ClusPath over other approaches (such as co-clustering) is that i) it joins the temporal and descriptive features in the same objective function and ii) it combines into the same optimization procedure the descriptive-temporal construction of the prototypes with the inference of the relations between clusters. The major advantage over constructing each component sequentially (like in TDCK-Means with *a posteriori* graph structure construction) is that the content of a cluster and relations between clusters influence each other during the optimization process. ClusPath is based on a "slow changing world" hypothesis, which assumes that changes in the population are gradual and "smooth". This hypothesis holds for many application domains, *e.g.*, scientific discussion topics, online communities *etc.* In applications in which this hypothesis does not hold (*e.g.*, stock market transactions, in which it is desirable to detect sudden changes), ClusPath can still be used, by lowering the degree in which this hypothesis is enforced.

Future work. We are currently experimenting with applying the algorithm to other applications, *e.g.*, detection of social roles in social networks, by passing through temporal behavioral roles. A social role is defined as a typical succession of behavioral roles. Another direction of research is describing the clusters with an easily comprehensible description by introducing temporal information into an unsupervised feature construction algorithm. Finally, it would be useful to compare the solution constructed by ClusPath with those issued by algorithms for detecting trajectories of moving clusters (*e.g.*, Kalnis et al (2005)).

6 Compliance with Ethical Standards

Conflict of Interest: The authors declare that they have no conflict of interest.
Research involving Human Participants and/or Animals: The authors declare that no part of the research presented in this manuscript involved any humans or animals.

Acknowledgements NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

References

(2016) Supplementary material: A temporal-driven clustering solution to inferring typical evolution paths. `http://goo.gl/KCWrSM`

Araujo R, Kamel MS (2014) Semi-supervised Kernel-Based Temporal Clustering. In: International Conference on Machine Learning and Applications, IEEE, ICMLA '14, pp 123–128

Armingeon, Klaus, Christian Isler, Laura Knöpfel DW, Engler S (2011) Comparative Political Data Set 1960-2009. University of Berne.

Chakrabarti D, Kumar R, Tomkins A (2006) Evolutionary Clustering. In: International Conference on Knowledge Discovery and Data Mining, ACM, SIGKDD '06, pp 554–560

Chi Y, Song X, Zhou D, Hino K, Tseng BL (2007) Evolutionary Spectral Clustering by Incorporating Temporal Smoothness. In: International Conference on Knowledge Discovery and Data Mining (KDD), San Jose, USA, pp 153–162

De la Torre F, Agell C (2007) Multimodal Diaries. In: Multimedia and Expo, IEEE, pp 839–842

De Smet Y, Eppe S (2009) Multicriteria Relational Clustering: The Case of Binary Outranking Matrices. In: Evolutionary Multi-Criterion Optimization, vol 5467, pp 380–392

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. Evolutionary Computation 6(2):182–197

Dunn JC (1973) A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. Journal of Cybernetics 3(3):32–57

Erixon L (2000) A Swedish Economic Policy: The Theory, Application and Validity of the Rehn-Meidner Model. Tech. rep., Department of Economics, Stockholm University

Gaffney S, Smyth P (1999) Trajectory clustering with mixtures of regression models. In: International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, USA, SIGKDD '99, pp 63–72, DOI 10.1145/312129.312198

Halsall-Whitney H, Thibault J (2006) Multi-objective optimization for chemical processes and controller design : Approximating and classifying the pareto domain. Computers & Chemical Engineering 30(6-7):1155–1168

Kafafy A, Bounekkar A, Bonnevay S (2011) A hybrid evolutionary metaheuristics (HEMH) applied on 0/1 multiobjective knapsack problems. In: Genetic and Evolutionary Computation, ACM Press, New York, USA, GECCO '11, p 497

Kalnis P, Mamoulis N, Bakiras S (2005) On Discovering Moving Clusters in Spatio-temporal Data. In: Bauzer Medeiros C, Egenhofer M, Bertino E (eds) Advances in Spatial and Temporal Databases, Lecture Notes in Computer Science, vol 3633, Springer Berlin Heidelberg, chap 21, pp 364–381

Liang Z, Tomioka R, Murata H, Asaoka R, Yamanishi K (2013) Quantitative Prediction of Glaucomatous Visual Field Loss from Few Measurements. In: International Conference on Data Mining, ICDM '13, pp 1121–1126

Lin WH, Hauptmann A (2006) Structuring continuous video recordings of everyday life using time-constrained clustering. In: Chang EY, Hanjalic A, Sebe N (eds) Multimedia Content Analysis, Management, and Retrieval, pp 60,730D–60,730D–9

MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Berkeley Symposium on Mathematical Statistics and Probability, vol 1, pp 281–297

Mihăiţă AS, Camargo M, Lhoste P (2014) Optimization of a complex urban intersection using discrete event simulation and evolutionary algorithms. In: International Federation of Automatic Control, IFAC'14, vol 19, pp 8768–8774

Rizoiu MA, Velcin J, Lallich S (2012) Structuring typical evolutions using Temporal-Driven Constrained Clustering. In: International Conference on Tools with Artificial Intelligence, IEEE, Athens, Greece, ICTAI '12, vol 1, pp 610–617

Rizoiu MA, Velcin J, Lallich S (2014) How to Use Temporal-Driven Constrained Clustering to Detect Typical Evolutions. International Journal on Artificial Intelligence Tools 23(04):1460,013

Rocha C, Dias LC, Dimas I (2013) Multicriteria Classification with Unknown Categories: A Clustering-Sorting Approach and an Application to Conflict Management. Journal of Multi-Criteria Decision Analysis 20(1-2):13–27

Sawaragi Y, Nakayama H, Tanino T (1985) Theory of multiobjective optimization, vol 176. Academic Press New York

Siddiqui ZF, Oliveira M, Gama J, Spiliopoulou M (2012) Where Are We Going? Predicting the Evolution of Individuals. In: Hollmén J, Klawonn F, Tucker A (eds) Advances in Intelligent Data Analysis V, Lecture Notes in Computer Science, vol 7619, Springer Berlin Heidelberg, pp 357–368

Wagstaff K, Cardie C, Rogers S, Schroedl S (2001) Constrained K-means Clustering with Background Knowledge. In: International Conference on Machine Learning, ICML '01, pp 577–584

Xu T, Zhang Z, Yu PS, Long B (2012) Generative models for evolutionary clustering. ACM Transactions on Knowledge Discovery from Data (TKDD) 6(2):7

Zitzler E, Laumanns M, Thiele L (2001) SPEA2: Improving the strength Pareto evolutionary algorithm. In: Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems, EUROGEN '01, pp 95–100

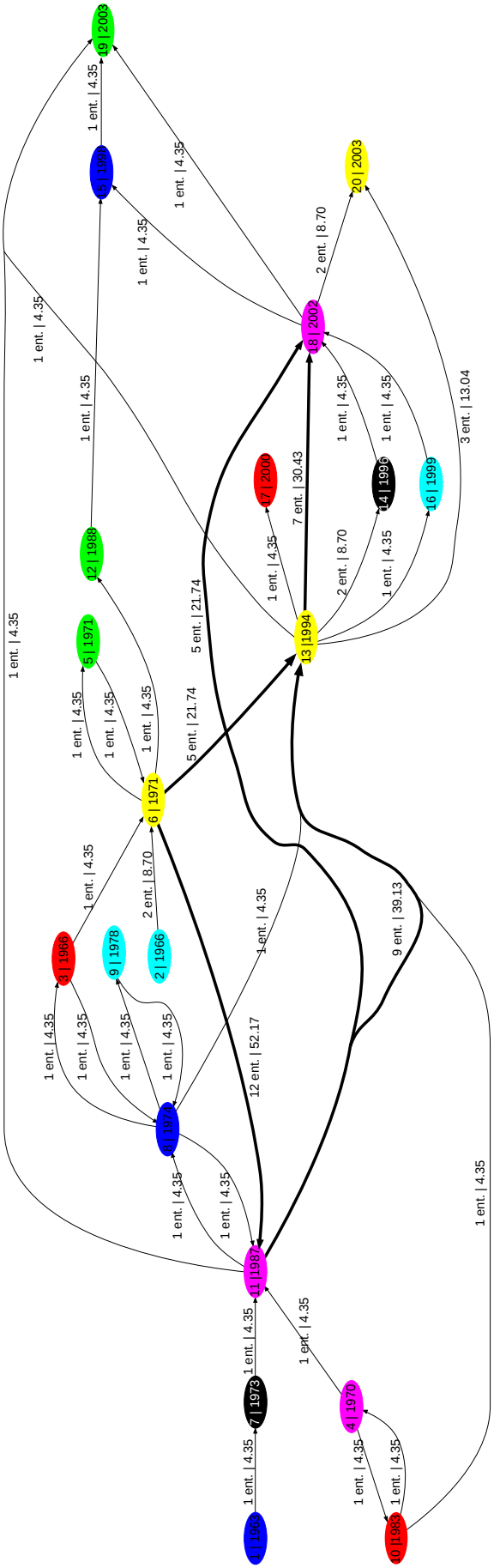


Fig. 7: Graph structure constructed *a posteriori* by TDCK-Means, on *Comparative Political Data Set I* with 20 clusters.